GETC

Be careful of buffer sizes and null termination when using getc()

Sean Barnum, Cigital, Inc. [vita[1]]

2007-03-22

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 5243 bytes

Attack Category	• Malicious Input
Vulnerability Category	• Input source (not really attack) • Buffer Overflow
Software Context	• String Management
Location	• stdio.h
Description	The getc() function is used to get the next character from the standard input stream. (The function returns the character read as an unsigned char cast to an int or EOF on end of file or error.) Other similar functions get the next character from other input streams (from files, for example). The getc() function, in isolation, is not a security risk. However, the function is often misused when filling buffers. Often, programmers will repeatedly call getc() and copy the characters into a buffer until a certain character is encountered, without checking the current position in the buffer. This can easily cause a buffer overflow. Also, it is easy to forget to include the null terminator at the end of the string in the buffer. An unterminated string can cause problems such as access violations.
APIs	**Function Name** / **Comments** fgetc getc getchar read
Method of Attack	If the attacker has control over the target buffer or the size of the target buffer or the input stream being read from, a buffer overflow condition can be induced.
Exception Criteria	When proper bounds checking is performed in the loop in which getc() is called repeatedly, then

1. http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html (Barnum, Sean)

	there is generally no security vulnerability with this function, per se.
Solutions	**Solution Applicability** / **Solution Description** / **Solution Efficacy** Any context where getc() is being used. If bounds checking is not present in the code and input is being derived from stdin, the vulnerability is especially high. / Explicit control of bounds checking. / Effective. Any context where getc() is being used. / Ensure that local strings derived from input streams are null terminated. / Effective.

Signature Details	int fgetc(FILE *stream);
	int getc(FILE *stream);
	int getchar(void);

Examples of Incorrect Code	`char str1[10];` `for(int i=0;i<15;i++)` `{` `str1[i]=getc(fp1);` `}` In this case, the system will read fifteen characters from the file pointed to by fp1 and overflow the buffer. `[...]` `int Byte, i;` `char HopeItFits[12];` `[...]` `i = 0;` `while((Byte = getc(stdin)) != ` `\n``)` `{` `HopeItFits[i] = Byte;` `[...]` `i++;` `}` `[...]` In this case, the user controls the vulnerability. As long as '\n' is not entered, the buffer can overflow.
Examples of Corrected Code	`[...]`

	int Byte, i;
	char HopeItFits[12];
	[...]
	i = 0;
	while((Byte = getc(stdin)) != `\n`)
	{
	HopeItFits[i] = Byte;
	[...]
	if(++i >= sizeof(HopeItFits))
	{
	fprintf(stderr, "Too much data read!\n");
	return(-1);
	}
	}
	[...]

Source Reference	http://www.linux-knowledge-portal.org/en/content.php?SEARCH&content/programming/secprog1.html[2]
Recommended Resource	
Discriminant Set	**Operating System**: • Any **Languages**: • C • C++

Cigital, Inc. Copyright

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. mailto:copyright@cigital.com